Modeling Controller Tasks for Safety Analysis*

Molly Brown and **Nancy G. Leveson**
Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{molly,leveson}@cs.washington.edu

Abstract

As control systems become more complex, the use of automated control has increased. At the same time, the role of the human operator has changed from primary system controller to supervisor or monitor. Safe design of the human–computer interaction becomes more difficult.

In this paper, we present a visual task modeling language that can be used by system designers to model human–computer interactions. The visual models can be translated into SpecTRM-RL, a blackbox specification language for modeling the automated portion of the control system. The SpecTRM-RL suite of analysis tools allow the designer to perform formal and informal safety analyses on the task model in isolation or integrated with the rest of the modeled system.

1 Introduction

Increased complexity of control systems and advances in computer technology have combined to give automation a more authoritative role in control systems. As a result, many of these control systems rely on both human and automated controllers. For these controllers to interact effectively, the human–computer interaction must be carefully.

We began looking at these issues while working with the Terminal Area Productivity (TAP) Project at NASA Ames [PPC97]. The TAP Project is designing terminal area procedures for air traffic using data links in addition to voice contact to communicate trajectories and routing information between the air traffic controller and the aircraft.

Traditionally, HCI design has focused on the human user's point of view: what functionality is needed to support the tasks the human must accomplish. There have been numerous models for human task analysis developed to identify the knowledge and steps required to perform each human task.

Unlike traditional task modeling methods that focus on analyzing specific aspects of the HCI design, such as steps to complete a task, user goals or knowledge representation, our modeling technique has a more general focus. Our technique allows the designer to model the steps required to complete the task so she can study how these steps interact with the rest of the system. Specific environment cues required to complete a task are modeled as conditions on transitions between steps in a task. Depending on the visualization created to inspect the model, the analyst can focus on the knowledge needed throughout the task, on the steps required to complete the task or on some other model aspect of interest.

Our modeling methodology takes a system-centric view compared to the human-centered view of other task analysis methods. With a tighter coupling between the human and automated controllers in complex systems, the human controller must be viewed as a part of the entire system, therefore the interaction between the human and the computer should be viewed in the context of the entire system.

Our modeling technique focuses formally analyzing a model of the controller's tasks independently and in the context of the complete control system model. Throughout the evolution of our method, we had three goals:

- To create a reasonable model of the actions of the human controller.
- To develop a model that can be formally analyzed with respect to safety concerns.
- To interface the model of the human controller tasks with formal models of the rest of the complext system.

*The research described has been funded by NAG-1-1894.

To realize these goals, we created a visual task modeling language that allows the analyst to easily represent the necessary information about the controller's tasks. These task models can be translated into the blackbox requirements specification language SpecTRM-RL and analyzed using a suite of analysis tools. SpecTRM-RL was developed to model all the components of control systems, therefore the SpecTRM-RL model of the controller tasks can be integrated with the model of the other components in the control system.

The rest of the paper is organized in the following way: Section 2 describes the SpecTRM tool suite and SpecTRM-RL modeling language; Section 3 describes our approach to achieving the goals we set for ourselves; Section 4 explains the example model that we use throughout the paper to illustrate our task modeling methodology; Sections 5 and 6 describe the visual task modeling language and how these models are created; Section 7 demonstrates how the safety analysis tools can be applied to these models; Section 8 discusses other work in the area of task modeling and analysis; and Section 9 discusses the contributions of this work and possible future questions to explore.

2 SpecTRM

The Software Safety Group at the University of Washington has developed a methodology for software specification called SpecTRM (Specification Tools and Requirements Methodology). Complex system development relies on multiple disciplines: system engineers, software engineers, human factors experts and application experts. SpecTRM takes a global system viewpoint to provide an environment to assist multidisciplinary teams. With a single consistent model of the complex system, analysts from each discipline can focus on the aspect of the system that is of interest to them.

The center of SpecTRM is SpecTRM-RL (SpecTRM Requirements Language), a formal requirements specification language for modeling blackbox behavior of control systems [Lev98]. SpecTRM-RL supports a wide-range of problem-solving strategies and tasks during system development and evolution.

Models written in SpecTRM-RL can be analyzed by the SpecTRM suite of tools. The SpecTRM tools were developed to detect errors and potentially hazardous behavior. The tool set is being expanded but currently includes:

- Model execution
- Automated formal analyses, such as consistency and completeness checks
- Automated tools to help with model exploration during forward and backward analyses
- Deviation analysis to test the robustness of system design to abnormal inputs
- Visualization tools to allow a wide-range of views of the executing model

These tools provide a flexible framework in which many complimentary analyses can be performed on a single model to help ensure total system safety.

3 Approach

We do not attempt to model erroneous human behavior, but limit our models to the expected controller behavior (both nominal and off-nominal) as defined in operational procedures. The models also include external inputs representing qualities of the environment that give rise to the human controller's decisions.

In many cases, the exact triggering conditions of a task are not necessary to have a meaningful model of the system. Any external or internal conditions that are particularly salient to the executing task are explicitly modeled while all other environmental effects are grouped into external conditions.

We quickly found that SpecTRM-RL models were not the most effective way to specify human tasks for system designers and human factors experts. For one thing, it was difficult to separate nominal from off-nominal behavior using the SpecTRM-RL notation. It was also difficult to see communication flow between the different components in the model. These limitations in expressibility led us to the development of a visual modeling language that has the characteristics lacking in SpecTRM-RL but remains easily translatable to SpecTRM-RL for analysis purposes.

4 Handoff Procedure Example

Throughout this paper, we will be using our model of a handoff procedure to illustrate our task modeling language. A "handoff", or a change of aircraft controller, occurs whenever an aircraft is changing from one controlling sector to another. The handoff procedure involves communication between the controller currently controlling the aircraft, the next controller to control the aircraft, and the pilot. Our model of this procedure includes the required tasks from each controller's point of view and the pilot's point of view. The handoff procedure model also includes a model of the radio used by the pilot to control the frequency to which she listens for controller communication. The radio component is included in the model because it

is an integral part of the hardware/software portion of the system during the handoff procedure.

We chose this model to demonstrate our modeling and analysis technique because the model is complex enough to have interesting characteristics while remaining clear enough to be easily understood by readers who are not familiar with air traffic control. The handoff procedure consists of multiple components each with variety of tasks to complete and interesting interactions with other components.

5 Description of Visual Modeling Language

Figures 1, 2 and 3 show the visual model for current implementation of the handoff procedure in air traffic control systems.

The key on Figure 1 displays the components of the visual modeling language. *States* are used to represent each step required to complete a higher-level task. To change from one state to the next in the model, a *transition* must occur. Changing the current model state represents the completion of one subtask and the beginning of the next subtask. An *event* is the triggering condition for a transition and an *action* results from completing a transition. The default, or start, state is denoted by an arrow head on the left side of the state. For example, in the Current Controller model of Figure 1 the state *Aircraft being controlled by current controller* is the start state. The model can transition to the state *Initiating handoff* if the event *Conditions for handoff occur* fires. In this example, there is no resulting action from the transition.

Color is used in the model to differentiate between events and actions. The events triggering a transition are shown in blue text above the transition, while any actions resulting from the transition are shown in red text beneath the transition. A green outline around event or action text denotes that this is a communication point between two entities in the model. For example, in Figure 1 there is a green outline around the action *Initiate Handoff* in the Current Controller model and a green outline around the event *Initiate Handoff* in the Next Controller model. The green outline denotes that the action of initiating a handoff in the Current Controller model causes a transition to occur in the Next Controller model.

Task positioning is used to represent relationships among the steps in a task. The normative actions are seen on the main horizontal axis of each controller's task model. Any non-normative behavior diverges from the main axis until the situation has been corrected and the normative procedure resumes. For example, in the Next Controller model of Figure 1, rejecting the handoff is not the expected or nominal behavior of the Next Controller. The events, states and actions required to handle this sequence of subtasks are shown below the main axis of the Next Controller's behaviors. When the off-nominal steps have been completed, the model can return to a state on the main horizontal axis to represent returning to the nominal behavior.

The visual modeling language also allows the system designer to represent the relationship among the tasks that are being carried out by the model entities. To accurately represent the load on the human controller, the system designer must understand these relationships accurately. The system designer must understand which tasks are sequential and which tasks can be performed in any order. The Pilot model, Figure 2, shows how these task relationships are represented. The action *Issue frequency change* must occur before the Pilot can perform the subtasks to change the radio frequency and read back the Next Controller's frequency to the Current Controller. The branching in the transition arrows shows how changing the radio frequency and performing the read back are executed in some undetermined order. These tasks could be performed in parallel or in a sequential order chosen by the Pilot. The system designer must understand these possible interactions to ensure that the tasks do not overload the Pilot no matter what order she chooses to execute the tasks.

6 Construction of Task Models

The basis of the visual model is a task analysis. The task analysis identifies the major tasks of the controller components in the system then breaks these high level tasks into subtasks, down to the level of the key presses, voice communications, display cues, etc. involved in performing the task. From the task analysis, the visual model is created. The visual model easily represents the temporal relationships among the tasks carried out by a single component in the model and the relationships among multiple components in the model.

The SpecTRM-RL model is created based on the visual model. The conversion is a straightforward process of converting the visual relationships to a text-based modeling language. Whereas the visual modeling language uses entities, states, events and actions to represent the system, the SpecTRM-RL model decomposes the system into components, operating modes, and input/output interfaces. A *component* is a portion of the system with a well-defined interface to the rest of the system. Each high-level system entity and

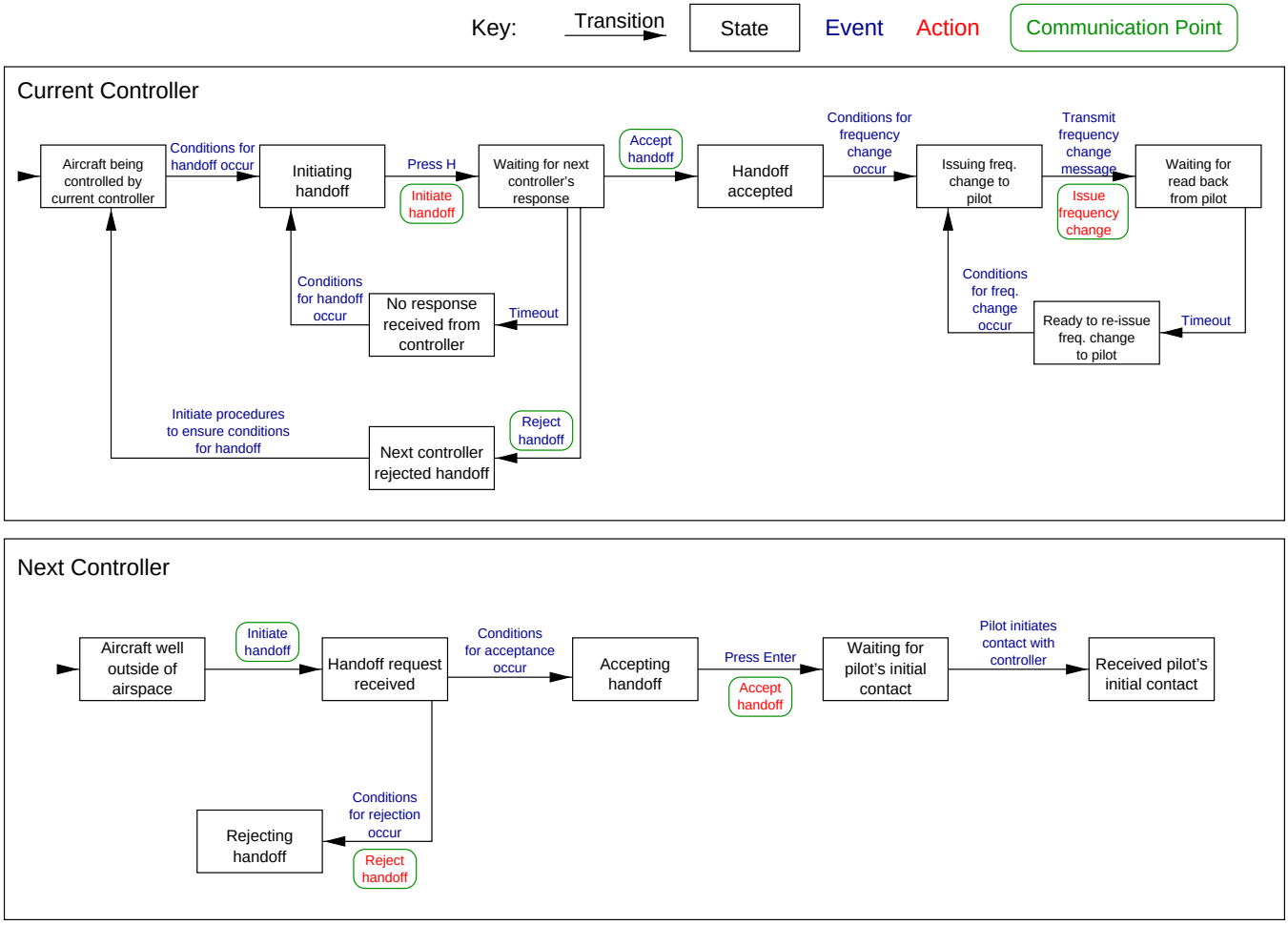


Figure 1: Model of Current Controller and Next Controller Tasks

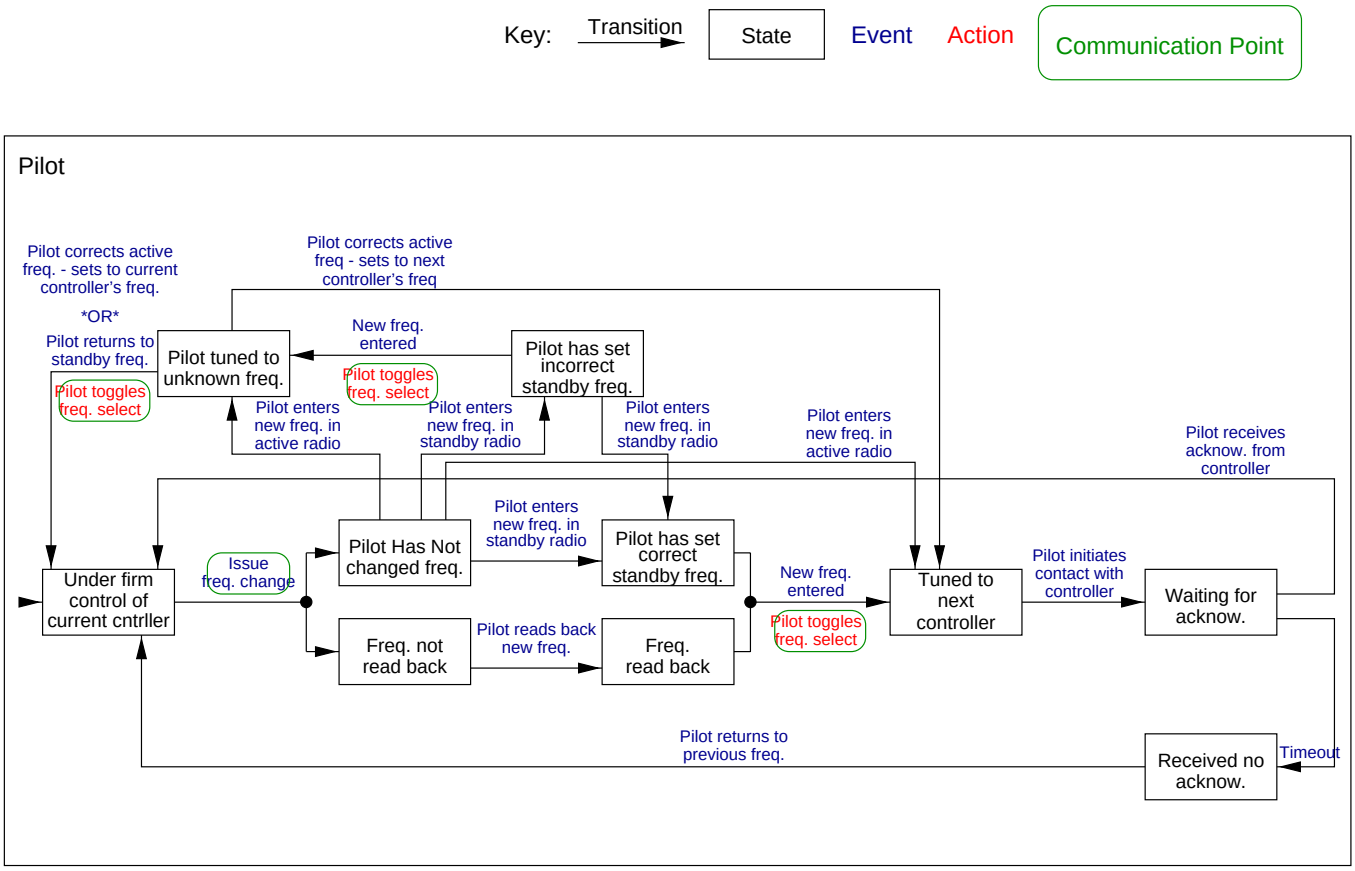


Figure 2: Model of Pilot Tasks

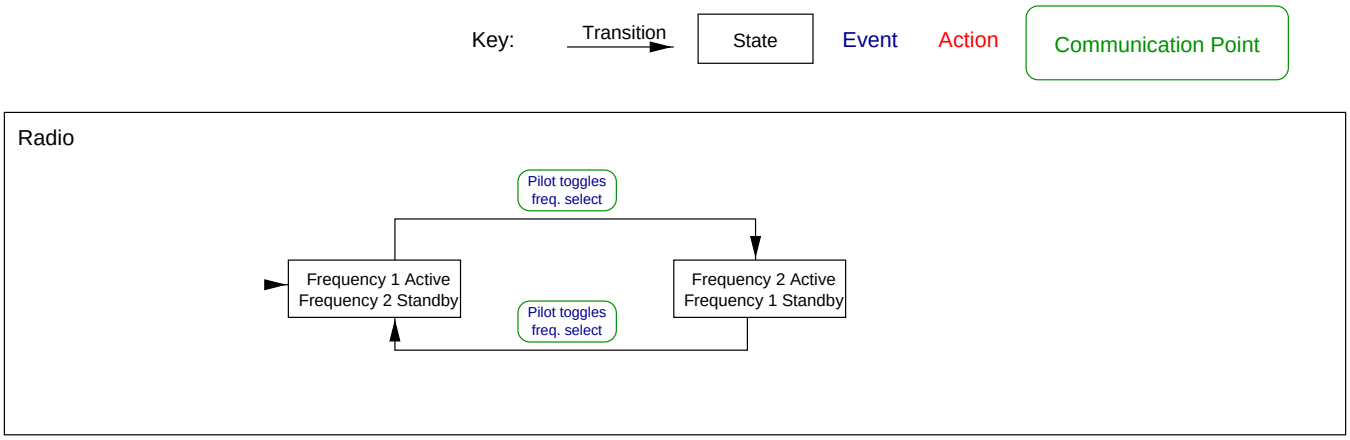
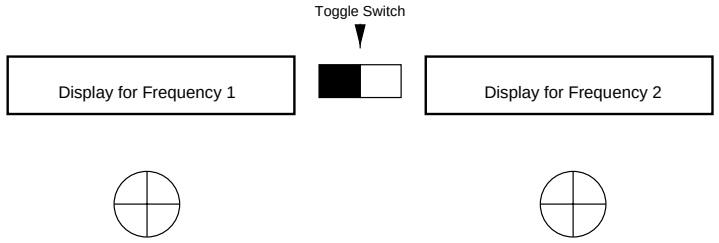


Diagram of Radio



Note: The radio had two displays and two dials. One display and dial monitors frequency 1 and the second display and dial monitors frequency 2. There is a toggle switch that allows the pilot to chose between frequency 1 and frequency 2 as the active frequency. The unselected frequency is referred to as the standby frequency.

Figure 3: Model of Radio

the external environment become a component in the SpecTRM-RL model. For the handoff procedure, the components in the SpecTRM-RL model are the Current Controller, the Next Controller, the Pilot, and the Radio. The states in the visual model translate to *operating modes* in the SpecTRM-RL model. For example, from the visual model shown in Figure 3 the Radio component has two operating modes: *Frequency 1 Active/Frequency 2 Standby* and *Frequency 2 Active/Frequency 1 Standby*.

Translating the communication interfaces is slightly more difficult. The events in the visual model map to input interfaces for the SpecTRM-RL model components. If the event was marked as a communication point with another entity in the visual model (denoted by the green outline around the event label), the corresponding input will come from another SpecTRM-RL component. Otherwise, the event will map to an input from the external environment. Actions in the visual model translate to the output interfaces for the SpecTRM-RL components.

7 Analysis of Models

The human procedure model, along with SpecTRM-RL models of the other parts of the system, can be used in the safety analysis of the human–computer interaction. The SpecTRM tool set currently allows model execution, various types of safety analysis, and visualization.

7.1 odel xecution

Because the task models are executable, the system designer can inspect the specified dynamic interaction between the system components, including both the operators and the automated components. In this way, procedural errors, possible inconsistencies in the procedures, or incomplete procedural specifications can be detected. We found that model execution helped us to find several errors in the specification.

7.2 Safety Analysis Tools

The SpecTRM tool set has multiple fully automated or partially automated analyses that can be performed on the models to help identify possible unsafe aspects of the system requirements. Consistency and completeness analysis identifies inconsistencies in the specification and conditions not accounted for in the specification [HL96]. For example, the automated completeness check on this model found that the Pilot's behavior is not completely specified. During the handoff procedure, the model does not show how the Pilot should act if she reads back an incorrect frequency. This incompleteness in the specification is likely to be the result of an oversight of the system designer as she builds the model, but the completeness check aids the analyst by highlighting these possible oversights.

Deviation Analysis provides a way to evaluate the specification for robustness against incorrect inputs [RL97a, RL97b]. The analyst denotes potential hazardous outputs that she wants to check for and hypothesizes deviations in the inputs, for example, that measured speed is lower than actual speed. The Deviation Analysis tool will determine whether the deviation can lead to a hazardous state and, if not, whether the hypothesized deviation plus other conditions could lead to a hazard.

Our Backward Analysis tool allows the system designer to start from a hazardous state and work backward to determine if and how that state could be reached. Critical points in the design are identified that can be modified to avoid the hazardous state. For example, a backward analysis of the handoff procedure model found that the Pilot model can reach the hazardous state *Pilot Tuned To Unknown Frequency*.

7.3 Visualization Tools

Our IBToolKit (Interface Builder Tool Kit) allows the system designer to create visualizations that can be linked to a SpecTRM-RL model [Pin97]. In addition to simply showing the results of the model executing in a visual format, visualizations can be created that highlight specific system qualities. These types of visualizations can aid in the design of automated systems to maximize the strengths of both the human and automated controllers.

One example of a visualization that we created for the handoff procedure is shown in Figure 4. In this visualization, the cognitive demands on the pilot during task execution are highlighted as the model executes. The states *Pilot has not changed frequency* and *Frequency not read back* are highlighted in red to denote the current state of the model. From this visualization, the system designer can see that the pilot had to detect a change in her environment in order to transition out of the *Under firm control of current controller* state.

Another possible visualization would be a display of the cockpit. As the portions of the tasks are completed by the pilot interacting with the cockpit controls, the corresponding areas of the cockpit could be highlighted. This visualization assists the system designer in determining whether the actions involved in the task support the cognitive processes required of the pilot.

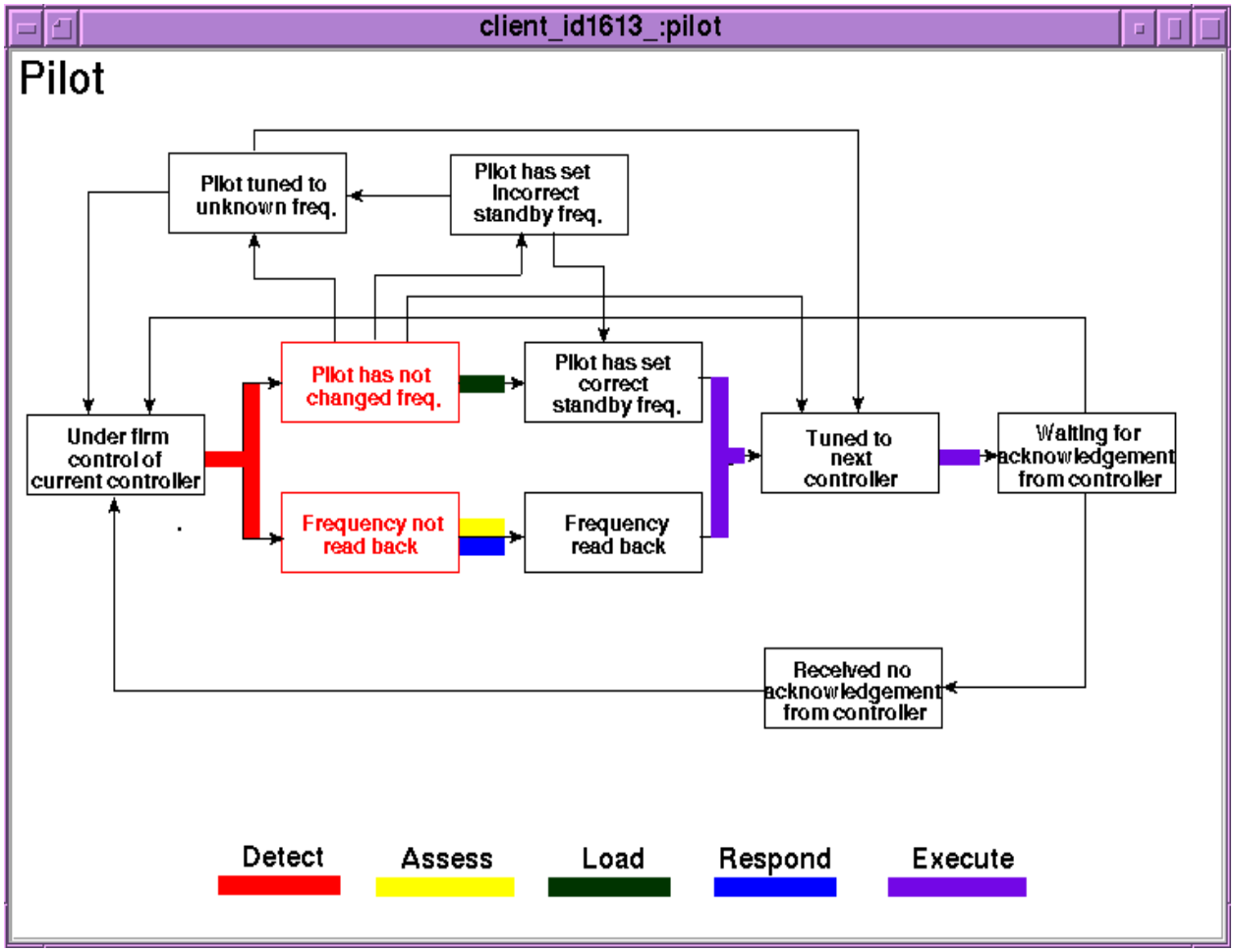


Figure 4: Visualization of cognitive demands on pilot during the handoff procedure.

8 Related Work

Task analysis models such as Hierarchical Task Analysis (HTA) [AD67] focus on steps required by the human to complete the given task. In HTA, tasks can be broken down into subtasks and there is no restriction on the level of decomposition. The view is very human-centered since this model was initially proposed to represent tasks for the purpose of training new users of applications. The focus is on the steps that must be completed by the human to accomplish a task without consideration for the operations of the computer. HTA is very flexible, but this flexibility often leads to ambiguity and inconsistency among models.

Many task models have been based on the Goals, Operators, Methods and Selection Rules (GOMS) framework [CMN83]. In the GOMS methodology, the user's goals are decomposed into subgoals that have operators (behaviors) and methods (sequences of behaviors) associated with them. The goals are decomposed into four different levels of description. This model focuses on error-free performance and models the tasks down to a very detailed description of the keystroke level of behavior.

Task Analysis for Knowledge Descriptions (TAKD) takes a different approach to task modeling by focusing on the knowledge required by people to complete the tasks [DJ89]. The TAKD method allows the knowledge needed to complete a task to be represented by a multilevel abstraction that traces the high-level knowledge down to the low-level steps necessary to complete the task.

and A. Whitefield, editors, *Cognitive ergonomics and human-computer interaction.* Cambridge University Press, 1989.

[HL96] M.P.E. Heimdahl and N.G. Leveson. Completeness and consistency analysis of state-based requirements. *Transactions on Software Engineering*, June 1996.

[JJ91] H. Johnson and P. Johnson. Task knowledge structures: psychological basis and integration in to system design. *Acta Psychologica*, 78(1-3):3–26, 1991.

[Lev98] N.G. Leveson. The SpecTRM-RL language. In preparation, 1998.

[Mit96] C.M. Mitchell. Task-analytic models of human operators: Designing operator-machine interaction. Technical report, Georgia Institute of Technology, 1996.

[Pin97] L.D. Pinnel. Visualizing requirement specifications: A toolkit for rapid prototyping of interfaces. Ph.D. qualifying project report, 1997.

[PPC97] T. Prevot, E. Palmer, and B. Crane. Flight crew support for automated negotiation of descent and arrival clearances. Technical report, NASA Ames Research Center, 1997.

[RL97a] J.D. Reese and N.G. Leveson. Software deviation analysis. In *International Conference on Software Engineering*, May 1997.

[RL97b] J.D. Reese and N.G. Leveson. Software deviation analysis: A safeware technique. In *AIChe 31st Annual Loss Prevention Symposium*, March 1997.

[SW95] N.D. Sarter and D. Woods. How in the world did I ever get into that mode?": Mode error and awareness in supervisory control. *Human Factors*, 37:5–19, 1995.

[YGS89] R.M. Young, T.R.G. Green, and T. Simon. Programmable user models for predictive evaluation of interface designs. In *Conference on Human Factors in Computing Systems (CHI '89)*, pages 15–19, May 1989.

[Yos96] G.R. Yost. Implementing the Sisyphus-93 task using Soar/TAQL. *International Journal of Human-Computer Studies*, 44(3-4), 1996.